Internet-based 3D PET Image Reconstruction using A Beowulf PC Cluster

‡D Shattuck, ‡J. Rapela, ‡E. Asma, ⋆A. Chatziioannou, †J. Qi, ‡R. Leahy
‡Signal and Image Processing Institute, Univ. of Southern California, Los Angeles, CA 90089
⋆Crump Institute for Molecular Imaging, University of California, Los Angeles CA90095
†Center for Functional Imaging, Lawrence Berkeley National Laboratory, Berkeley, CA 94720

Abstract— **We describe an approach to fast iterative reconstruction from fully 3D PET data using a network of PentiumIII PCs configured as a Beowulf cluster. To facilitate the use of this system, we have developed a browser-based interface using Java. The system compresses PET data on the user's machine, sends this data over a network, and instructs the PC cluster to reconstruct the image. The cluster implements a parallelized version of our preconditioned conjugate gradient method for fully 3D MAP image reconstruction. We report on the speed-up factors using the Beowulf approach and the impacts of communication latencies in the local cluster network and the network connection between the user's machine and our PC cluster.**

Keywords— **3D PET, Beowulf cluster, distributed computing, iterative reconstruction**

I. Introduction

Iterative reconstruction of clinical PET images using statistically optimal algorithms can require an hour or more of computation on a single-processor computer for fully 3D data sets. Dramatic reductions in computation time have been achieved by converting fully 3D data sets to 2D using rebinning algorithms and then using iterative 2D reconstruction methods [1]. Further reductions have been achieved using the ordered-subsets EM (OSEM) algorithm which can achieve acceptable results in just a few passes through the data. However, these speed-ups are achieved at a price: the rebinning methods, even if exact for true line integrals, are unable to accurately model the true physical response of the scanner. Similarly, the OSEM method never optimizes the likelihood objective function and the results can be highly dependent on the number of subsets and number of iterations that are used. In our work [3], [2] we have concentrated on using convergent algorithms to compute maximum a posteriori (MAP) or equivalently penalized-ML solutions to the PET reconstruction problem. The more accurate models that we use with fully 3D data sets have been shown to improve image resolution [3] but inevitably lead to longer computation times.

Our approach to reducing reconstruction time is to use rapidly converging methods such as the preconditioned conjugate gradient method. Further reductions can be obtained using multiprocessor computing. Previously we have used multithreading methods to parallelize the code across multiple CPUs in a single symmetric multiprocessor (SMP) server. This arrangement is attractive since the servers typically have shared memory and hence there is minimal overhead incurred in distributing data across the processors. In tests with a four processor server, we were able to achieve speed-up factors of approximately 3.4 relative to a single processor. Unfortunately, the number of processors in standard servers is usually limited to four and the cost is high relative to single or dual processor systems. For this reason we have recently investigated the use of a Beowulf PC cluster that allows us to use a large number of low cost systems to achieve substantial speed-up relative to a single computer. Vollmar et al [5] recently reported the use of a PC cluster for 3D PET reconstruction. Their approach differs from that described here in that the forward and backprojection were based on on-the-fly computation rather than a precalculated system matrix. Similarly Labbe et al [6] present a set of forward and backprojection operators suitable for cluster and parallel computing but again these are based on on-the-fly computation.

The Beowulf cluster is simply a network of Unix or Linux workstations. For the purposes of code parallelization, the cluster is configured with a head-node that controls the program and a set of worker-nodes that handle processes spawned by the head node. The difference between the Beowulf cluster and a multiple CPU server is that the former do not have shared memory, and data must be transferred via a local ethernet between processors. This is often the bottleneck in performance of these clusters and of particular importance in PET image reconstruction where the data sets and image volumes are large. Here we report on our progress using a combination of multithreading and distributed computing on a Beowulf cluster consisting of 9 dual 933MHz PentiumIII computers connected via a 100mb/s switched ethernet.

A second goal of our work was to decouple the computer used for reconstruction from that used to acquire data. To do this we have developed a web-browser based interface to our distributed computing code using Java. Thus data can be processed using the cluster from any computer connected to the Internet. While data transfer may be slow for standard Internet connections, the availability of Internet2 connections at many research facilities make this approach viable. We report on an experiment we have performed by reconstructing data residing on the PET system computers in the Nuclear Medicine clinic at UCLA using the cluster at the Signal and Image Processing Institute at USC.

This work was supported by the National Cancer Institute under Grant No. R01 CA59794 and the National Foundation for Functional Brain Imaging.

II. Methods

A. MAP Image Reconstruction

We use a MAP estimation algorithm to reconstruct PET images [3]. In this approach, the data are modeled as:

$$\overline{\mathbf{y}} = \mathbf{P}\mathbf{x} + \overline{\mathbf{r}} + \overline{\mathbf{s}} \tag{1}$$

where $\overline{\mathbf{y}}$ is the mean of the data, $\mathbf{x}$ is the source distribution, $\overline{\mathbf{r}}$ is the mean of the randoms, and $\overline{\mathbf{s}}$ is the mean of the scattered events. $\mathbf{P}$ is the system matrix describing the probability that an event is detected, which we factor as:

$$\mathbf{P} = \mathbf{P_{norm}}\mathbf{P_{blur}}\mathbf{P_{attn}}\mathbf{P_{geom}} \tag{2}$$

where $\mathbf{P_{geom}}$ is the geometric projection matrix describing the probability that a photon pair reaches the front faces of detector pair in the absence of attenuation and assuming perfect photon pair colinearity, $\mathbf{P_{blur}}$ models photon pair non-colinearity, inter-crystal scatter and crystal penetration, $\mathbf{P_{attn}}$ contains attenuation correction factors for each detector pair, and $\mathbf{P_{norm}}$ is a diagonal matrix containing the normalization factors.

Reconstructions are computed as the maximizer of a posterior probability equal to the sum of the log-likelihood of the data, $\mathbf{y}$, conditioned on the image, $\mathbf{x}$, and the log-prior, which has the form of a Gibbs energy function:

$$\begin{aligned} \ln\ p(\mathbf{y}|\mathbf{x}) &= \textstyle\sum_i \{\overline{y}_i + y_i \ln(\overline{y}_i)\} \\ &- \textstyle\sum_j \sum_{\substack{k\in\mathcal{N}_j \\ k>j}} \kappa_{jk} V(x_j - x_k) \end{aligned} \tag{3}$$

where $V(x)$ is the potential function.

As in our previous work on PET image reconstruction, a preconditioned conjugate-gradient algorithm was used for optimization. In particular, the following preconditioned Polak-Ribiere form of conjugate gradient method was used.

$$\mathbf{x}^{(n+1)} = \mathbf{x}^{(n)} + \alpha^{(n)}\mathbf{s}^{(n)} \tag{4}$$

$$\mathbf{s}^{(n)} = \mathbf{d}^{(n)} + \beta^{(n-1)}\mathbf{s}^{(n-1)} \tag{5}$$

$$\mathbf{d}^{(n)} = \mathbf{C}^{(n)}\mathbf{g}^{(n)} \tag{6}$$

$$\beta^{(n-1)} = \frac{(\mathbf{g}^{(n)} - \mathbf{g}^{(n-1)})'\mathbf{d}^{(n)}}{\mathbf{g}^{(n-1)'}\mathbf{d}^{(n-1)}} \tag{7}$$

The PCG algorithm is initialized with $s^{(0)} = d^{(0)}$ and iteratively computes the conjugate directions. It it necessary to check that $s^{(0)}$ is an ascent direction. In the case that $s^{(n)'}g^{(n)} < 0$, $s^{(n)}$ is a descent direction and the algorithm is reinitialized with $s^{(n)} = d^{(n)}$. The step size, $\alpha^{(n)}$, is computed at each iteration using a Newton-Raphson line search to maximize the objective function. We incorporate a positivity constraint by using a bent-line search as we describe in [3].

Here we report on application of this algorithm, using the PC cluster, to data collected in 3D mode using the CTI ECAT HR+ scanner. The data from the HR+ scanner was a standard 3D dataset rebinned with a span of 9 and a maximum ring difference of 22. There were 239 sinograms each of size 288 (elements) by 144 (angles) giving a total emission sinogram size of 40 MB. Attenuation correction requires a second sinogram of the same size, thus the total data for a single frame is on the order of 80MB in size. These file sizes are important when considering the impact of reconstruction via a browser over an internet connection. The other files that are required are either small (such as the factored normalization file) or can be stored on the cluster (such as the forward projection matrix, $\mathbf{P}$). Voxel sizes used in our reconstructions were $2.25mm \times 2.25mm \times 2.42mm$ for the HR+ and images were of size $128 \times 128 \times 63$ so that the image is of size 4MB when saved as 4-byte real values. Thus, transfer times for the reconstructed images back to a remote user are small compared to those for sending the data to the cluster.

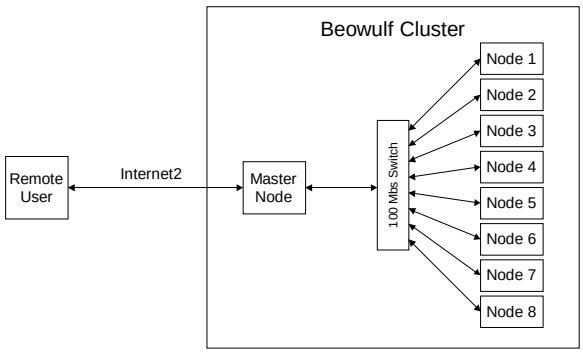

Fig. 1. Architecture of the PC cluster.

B. Beowulf Clusters and Code Parallelization

B.1 System setup

We built a Beowulf cluster consisting of one master node and eight worker nodes. Each worker node is a rackmounted dual processor Intel Pentium III 933 MHz system with 512MB of RAM and 20GB of disk space. The master node is also a rackmounted dual processor Intel Pentium III 933 MHz system, but has 1GB of RAM and a 36GB hard drive. The master node has dual network interface cards, allowing the cluster to have a private network but still be accessible from the Internet. The configuration of the cluster is shown in Fig 1. We configured the system with the Linux operating system (RedHat v. 7.0; Linux kernel version 2.4). We also installed the Local Area Multicomputing (LAM) 6.5.1 version of the Message Passing Interface (MPI) software onto each node. MPI is an open standard for communicating data between computer processes; LAM is an implementation for use on clustered computers and provides a programming environment that is portable to other architectures.

B.2 Code parallelization

Analysis of our algorithm's performance on a single computer revealed that two operations dominated computation: the back projection of the sinograms into image space and the forward projection of the image into sinogram space. Computation of the gradient in (6) requires a forward and back projection; a second forward projection

is required prior to doing the line search to compute $\alpha^{(n)}$ in (4). We distributed the processing of these key operations across the cluster. Ignoring communication costs, we are able to achieve roughly a factor-$0.75 \cdot N$ speed-up on the forward and backward projections using N nodes of the cluster. This number is less than N because the workload is not perfectly balanced across the processors, but still represents a dramatic improvement in computing time. Were we to communicate the sinogram data during the iterations of the algorithm, the high cost of passing these results among the nodes would rapidly consume the gain in performance. Fortunately, we can decompose our problem such that the slave nodes never need to receive or transmit sinogram data once the iterations have begun.

The forward and back projection operators are both linear transformations, and represented as a factored system matrix as described above. During back projection, each element of the image may be a function of several elements of the sinograms. Each sinogram is transformed by the system matrix to contribute to the reconstructed image. We can partition this transformation based on arbitrary sets of sinograms, apply the system matrix separately to these sinograms to obtain their contribution to the reconstructed image, and then sum these partial results to obtain the entire transformation. In our distributed implementation, we assign each node a range of sinograms for which it is responsible. The node keeps updated versions of the data for these sinograms, and backprojects them into imagespace when requested by the head node. These results are sent back to the head node, where they are combined into a single image.

The forward projection problem can also be decomposed into functions producing individual sinograms; however, each operation will still require the full image that is being forward projected. Fortunately, the communication cost of transmitting images to each node is relatively small compared to the cost of transmitting sinogram data or performing the reconstruction computation. During forward projection, the head node broadcasts the image to each node; the nodes are responsible for producing the same sinograms that they will use during backprojection. During the iterations, the nodes generate any sinogram data they will need, and thus do not need to communicate their sinogram data to other nodes of the cluster. We distributed some additional computation to the nodes to eliminate the need to send any sinogram data to any other nodes; the effects of this distributed processing are small compared to the gains from distributing the projection operations.

A second layer of parallelization is used on each node, as each has dual processors. The projection problems are again decomposed based on sinograms, and two threads are spawned on each node to handle the projections. In this case, we achieve better load balancing as the node can dynamically assign sinograms to the threads as soon as they have finished processing. The computation of the image prior (3) is also multi-threaded on the head node. This operation may be distributed to the cluster in future work; however, this would require broadcasting of image vectors and the cost of communication may outweigh the benefit of distributed processing.

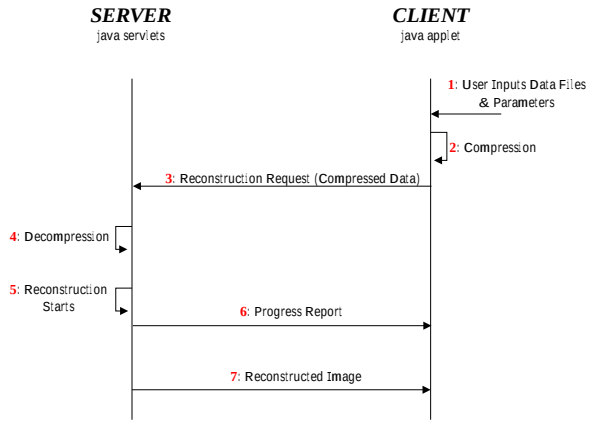

Fig. 2. Architecture of the web interface to the 3D PET reconstruction program. The user supplies the data files and parameters to the java applet. The applet compresses the data files and submits a reconstruction request to the server. The server decompresses the data files, starts the reconstruction and sends the reconstruction progress to the applet, which displays it in the client browser. When the reconstruction finishes the server sends the reconstructed image to the applet which displays the results in the client browser.

The projection routines are also used during initialization, so improvements to them will reduce start-up costs. Additionally, the geometry matrices used in projection can be hundreds of megabytes in size, and are needed on each node. These matrices are used repeatedly for a particular scanner and voxel size, thus we store copies of these files on the local hard drive. This reduces the network burden further.

C. Java Browser-based Interface

We developed a Java based interface to the 3D MAP reconstruction program that allows users across the Internet to run reconstructions on our Beowulf PC cluster. The interface consists of two components, a client module and a server module. Fig. 2 illustrates the architecture of the interface.

The client module was implemented as a Java applet and can run on standard web browsers. The user supplies the data files (emission file, normalization file, etc) and parameters (number of bed positions, number of frames, etc) for the MAP reconstruction and submits a reconstruction request. The client module collects the parameters and data files for the reconstruction and sends them to the server. The server module was implemented as a Java servlet. It receives the data from the client and starts the reconstruction. Text messages describing the reconstruction progress are sent to the client. The applet displays the reconstruction progress in the browser. When the reconstruction finishes the applet receives the reconstructed images and statistics about the reconstruction process.

The data sizes used in 3D PET reconstructions are large. The data size we used for HR+ reconstruction was 80MB per frame. To reduce transfer time the applet compresses

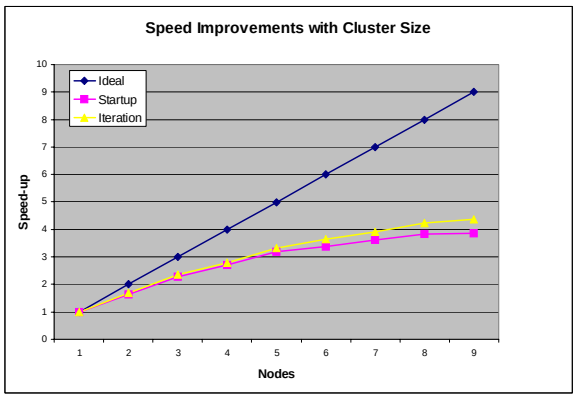

Fig. 3. Speed increase to the iterative portion of reconstruction code for different sized clusters.

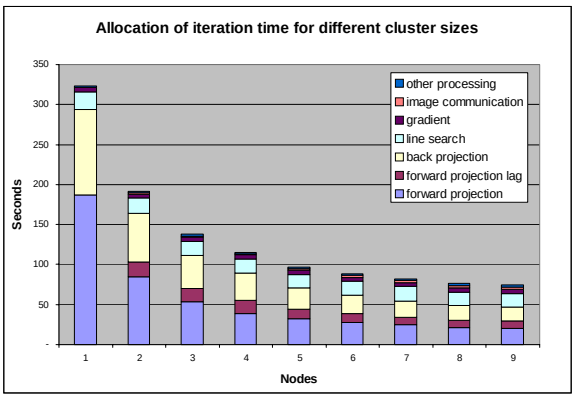

Fig. 4. Usage of computing time in a single iteration of reconstruction. Forward projection lagtime is the difference between when the head node finishes its portion of the forward projection and when all nodes are finished; some of this time may be used by the head node to perform additional computation.

the data files before sending the reconstruction request to the server side. The compressed data is transferred over the network and received by a servlet on the cluster. The servlet decompresses the data before starting the MAP reconstruction.

D. Network Connections

To evaluate performance over Internet2, the PC-cluster server was connected through a 100mb/s network to the University of Southern California backbone to Internet2. The client computer containing the data was similarly connected through the UCLA computer network. The link between USC and UCLA is part of the California Research and Education Network-2 (CalREN-2). CalREN-2 is a high-performance advanced-services network with a minimum communication bandwidth of 622Mbs.

III. Results

We performed reconstructions on our cluster using different numbers of nodes to assess the benefit of using distributed processing. Figure 3 shows the performance gains achieved on both the initialization and the main loop of the program for reconstruction of HR+ 3D data. The chart show that we achieved better than $N/2$ increases in processing speed, where N is the number of nodes in the cluster, for up to 8 nodes. Our performance begins to flatten with the 9th node, which reduced the iteration time of the reconstructions by a factor of 4.36. This represents a significant performance gain over using a single dual-processor machine. Figure 4 shows the key components of a reconstruction iteration for the HR+ data as computed with different sized clusters. Forward and back projection clearly dominate the computation time when the algorithm is performed on a single node. As the number of nodes increases, the times for these operations are greatly reduced. When the ninth node is added to the cluster, the line search requires almost as much time as the forward and back projection. This figure indicates that to achieve further gains by adding more nodes we must either distribute additional processing or perform better load balancing.

Additional overhead for data transfers over Internet2 were minimal. For example transfer of the combined transmission and attenuation correction file (80MB) in uncompressed format took 54 seconds. from UCLA to USC. Typically we can achieve 50-75% compression using the compression applet which will reduce the transfer time to between 27 and 13.5 seconds. However, the time taken to compress the two files is approximately 2 minutes on a 450MHz UltraSPARC workstation, which exceeds the transfer time required for the uncompressed file. For systems with slower Internet connections, the trade-off between compression and transfer times will be different and use of compression will be appropriate. This preliminary study demonstrates the feasibility of using remote PC-clusters for image reconstruction , particularly for PET sites with access to fast networks. Furthermore, the cluster presents a relatively low-cost approach to achieving practical reconstruction times in 3D iterative PET reconstruction.

References

[1] P. Kinahan, C. Michel, M. Defrise, D. Townsend, M. Sibomana, M. Lonneux, D. Newport, and J. Luketich: Fast iterative image reconstruction of 3D PET data, Proc. IEEE Nuclear Science Symposium and Medical Imaging, pp. 1918–1922, 1996.

[2] Mumcuoglu E.U., Leahy R. M.,Cherry S.R. and Zhou Z.: Fast gradient-based methods for Bayesian reconstruction of transmission and emission PET images. IEEE Trans. Med. Imag. **13**, (1994), 687–701

[3] J. Qi, R. M. Leahy, S. R. Cherry, A. Chatziioannou and T. H. Farquhar: High resolution 3D Bayesian image reconstruction using the microPET small-animal scanner. Phys. in Med. Biol. **43**, 1998, 1001-1013

[4] D. Becker and T. Sterling and D. Savarese and B. Fryxelland K. Olson: Communication Overhead for Space Science Applications on the Beowulf Parallel Workstation, Proc. High Performance and Distributed Computing, 1995.

[5] St. Vollmar, M. Lercher, C. Knvss, C. Michel, K. Wienhard and W.D. Heiss: BeeHive: Cluster Reconstruction of 3-D PET Data in a Windows NT network using FORE, Proc. IEEE Med Imag Conf, Lyons, Oct, 2000.

[6] C. Labbe, H. Zaidi, C. Morel, K. Thielemans: An object-oriented library incorporating efficient projection/backprojection operators for volume reconstruction in 3D PET, Proc. 3D99, Egmond aan Zee, Netherlands, pp 137-140, 1999.